# Python in Fluidity

Ralf D Perpeet
(Supervised by Dr David Ham)

Imperial College London

August 15, 2008

# Table of contents

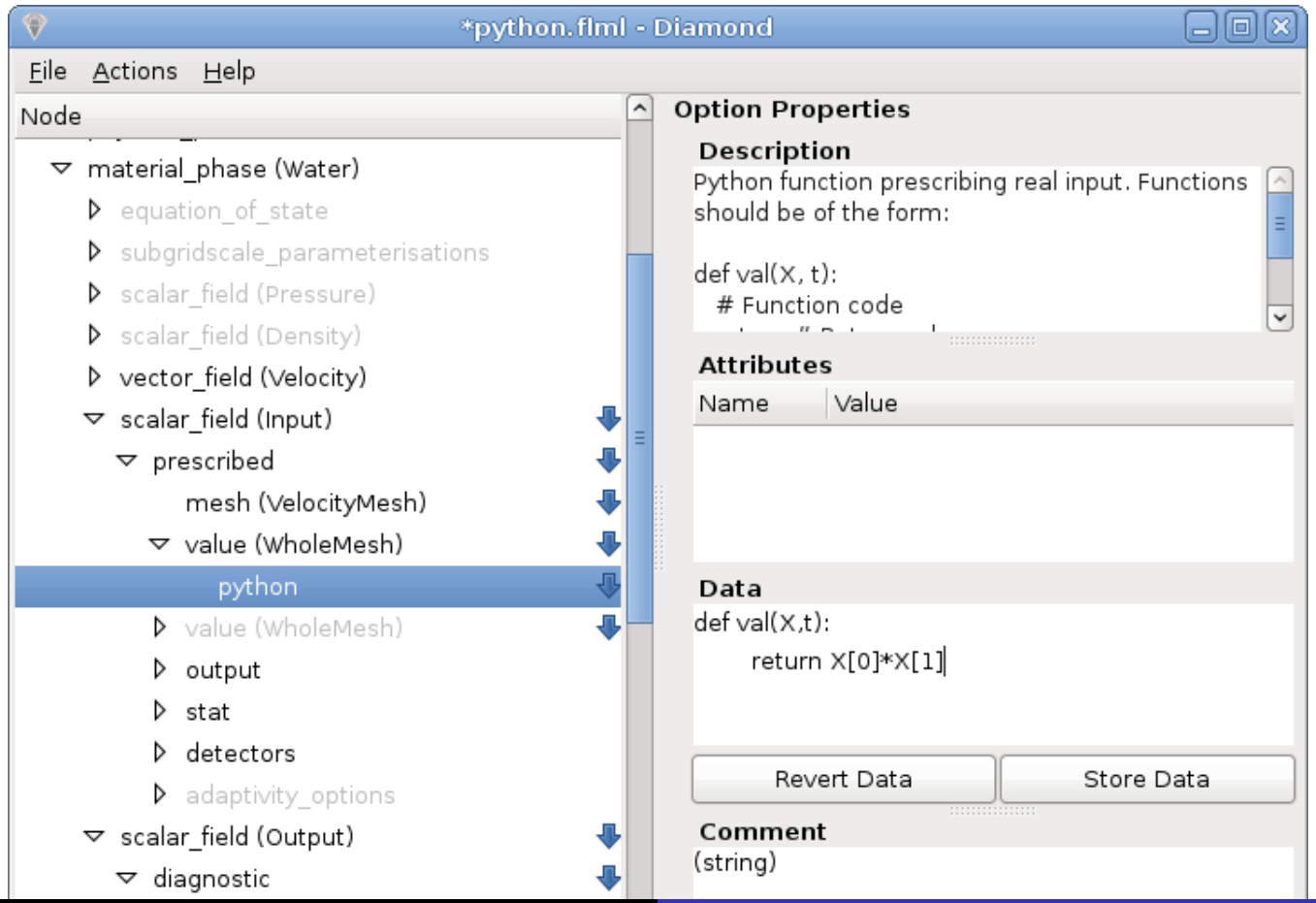
# Python in Fluidity

# Goals of This UROP

- Make Fluidity more accessible towards users' requirements by having a Python environment available
- Pass data structures from Fortran to Python in a memory efficient way
- Give some functionality of the Fortran code in the Python environment

# Previous Functionality

- set_from_python()
- Passed tuple to return the values for each node in the field



# Python in Fluidity

# Setup and Basics for the Python Environment

## Initialization and Finalization in *main.cpp*

- #include "python_statec.h" as external
- python_init_() (note the underscore!)
- python_end_()
- Call only once before and after the main program.
- export PYTHONPATH=fluidityroot/python/

## Cleaning Up and Resetting

- python_reset_()
- Clears all python dictionaries apart from the essential ones (__doc__, numpy, string, etc.)
- Exception: *persistent*{} - use this dictionary to keep data from being deleted

# Running Python Code in Fluidity

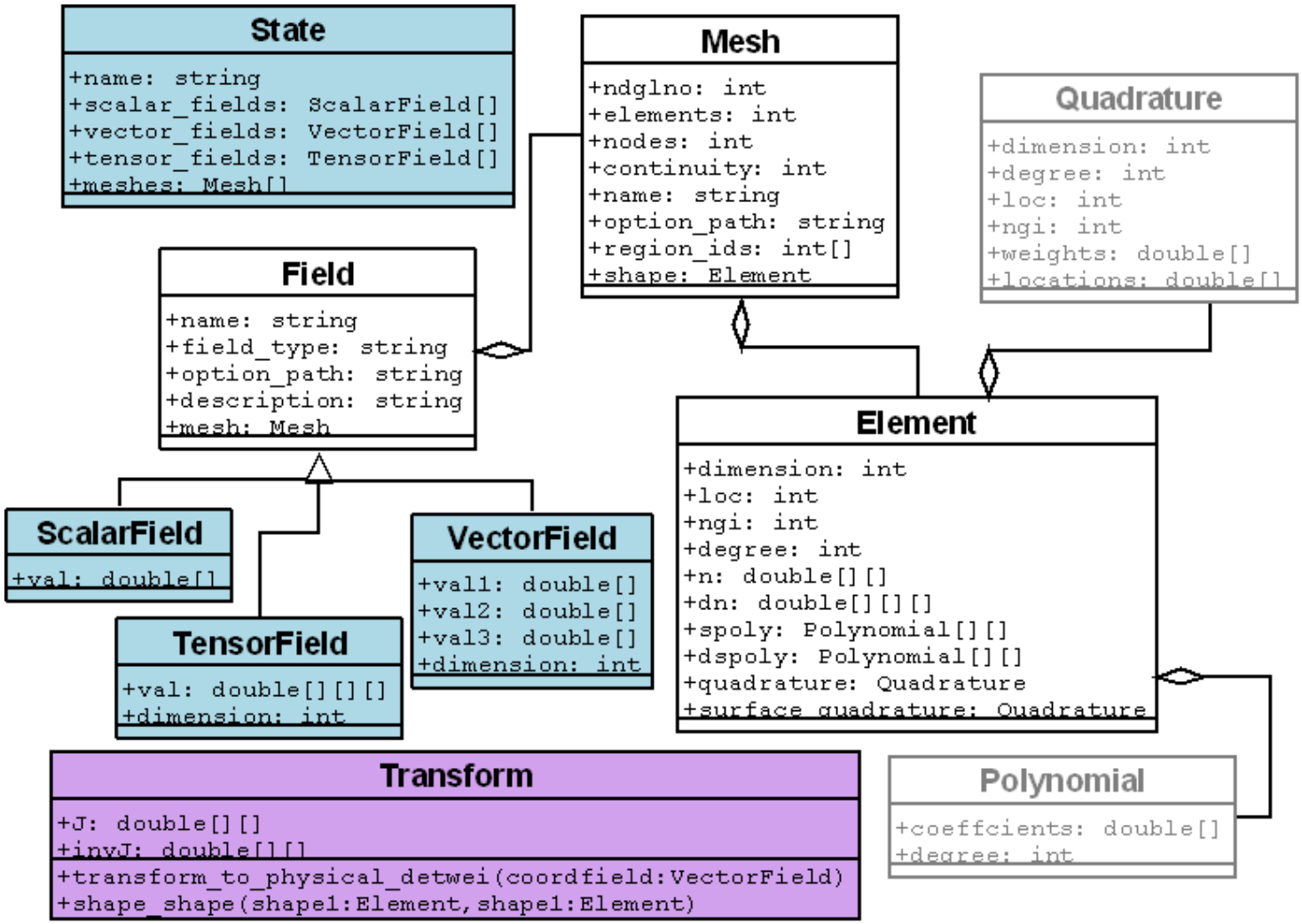
## In Fortran

- python_run_string(string *code*), e.g. *"a = 5; print a"*
- python_run_file(string *filename*) - Better to write a module and import via run_string() though
- python_add_state(state_type *State*)

## In Python

- Last added state always available as *state*
- All added states: *states*{'State1','MyState','Water'}
- Load modules from the ./python folder, e.g. *import fluidity.ocean_biology*
- A state has different attributes:
- scalar_fields, vector_fields, tensor_fields (passed by reference)

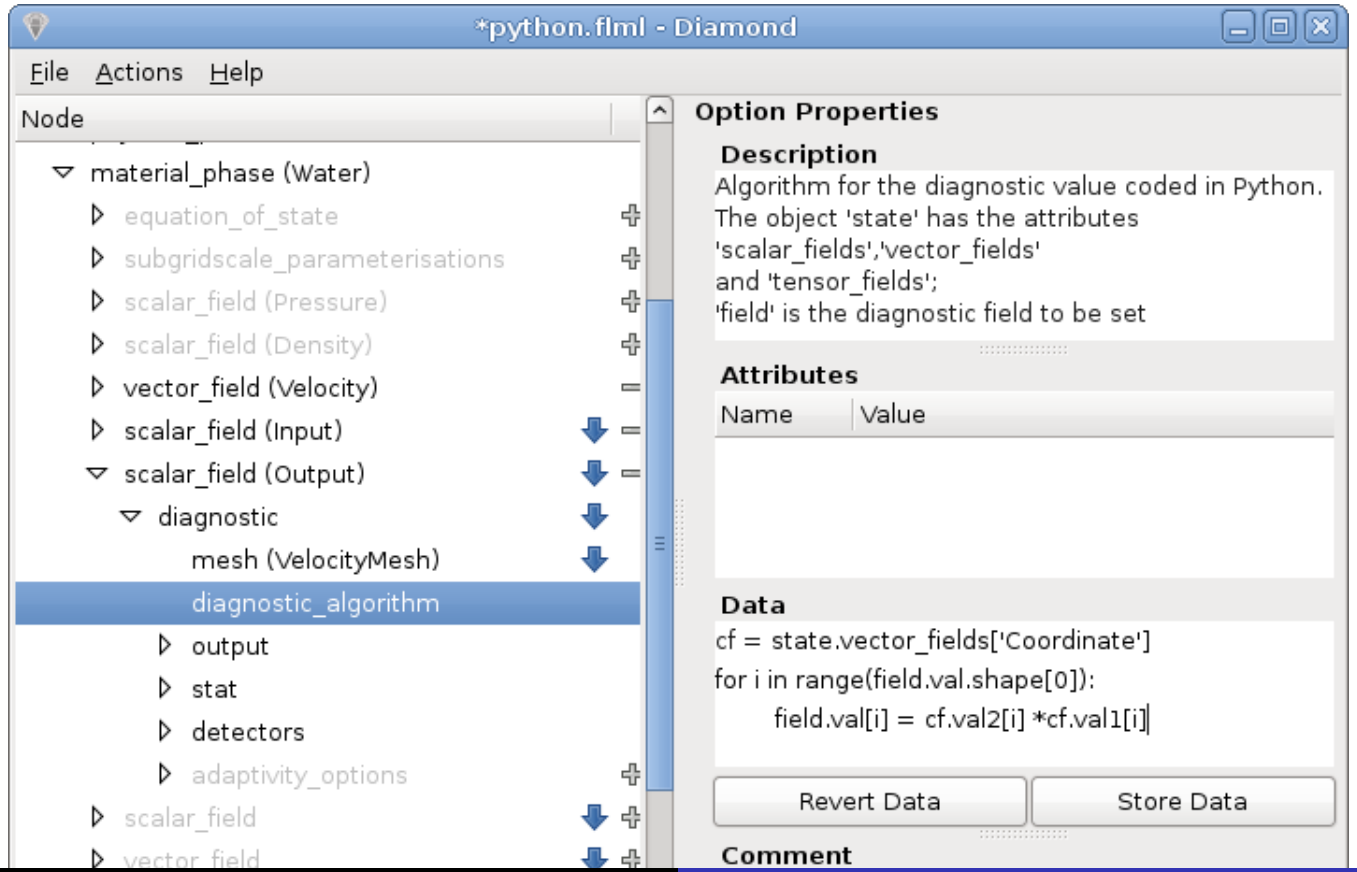
# Data Structures in state_types.py



# Set Diagnostic Fields From Python

## Diamond Functionality

- Scalar field to be set availabe as *field*
- All states available in *states*



# Noteworthy Functions From Fortran Available in the Python Interpreter

## Field Methods

- ele_val(int ele_number) - Return the values of field at the nodes of ele_number
- ele_val_at_quad(int ele_number) - Return the values of field at the quadrature points of ele_number

## Mesh Methods

- ele_nodes(int ele_number) - Return all nodes associated with the element ele_number

## Transform Methods

- transform_to_physical_detwei(Field field)
- grad(int gi,Element shape)
- shape_shape(Element shape1, Element shape2)

# Python in Fluidity

# Affected Files

## Fortran

- /femtools/python_state.F90
- /assemble/Diagnostics_fields_wrapper.F90

## C/C++

- /main.cpp
- /include/python_statec.h
- /femtools/python_statec.c

## Python

- /python/fluidity/state_types.py

## Set a Diagnostic Field From Python

```
subroutine set_diagnostic_field_from_python(state,field)
type(state_type),intent(inout) :: state
type(scalar_field), intent(inout) :: field
character(len=PYTHON_FUNC_LEN) :: pycode
#ifdef HAVE_NUMPY
! Clean up to make sure that nothing else interferes
call python_reset()
call python_add_state(state)
call python_run_string("field =
state.scalar_fields[''//trim(field%name)//'']")
call
get_option(trim(field%option_path)//"/diagnostic/diagnostic_algorithm",pycode)
call python_run_string(trim(pycode))
#else
FLAbort("Python diagnostic fields require NumPy, which cannot be
located.")
#endif
end subroutine set_diagnostic_field_from_python
```

# Passing one of the fields (Fortran)

## python_add_scalar()

```
interface
subroutine
python_add_scalar(sx,x,name,nlen,field_type,option_path,oplen,state_name,snlen,&
&mesh_name,mesh_name_len)
integer :: sx,nlen,field_type,oplen,snlen,mesh_name_len
real, dimension(sx) :: x
character(len=nlen) :: name
character(len=snlen) :: state_name
character(len=oplen) :: option_path
character(len=mesh_name_len) :: mesh_name
end subroutine python_add_scalar
end interface
```

# Passing one of the fields (C)

## python_add_scalar_()

void python_add_scalar_(int *sx, double x[],char *name,int *nlen, int *field_type, char *option_path,int *oplen, char *state_name,int *slen, char *mesh_name, int *mesh_name_len)

- Note the _ , appended to every function name from Fortran in C
- All values passed by reference (pointers!)
- Fortran strings are not null-terminated (call char* fix_string(char *s, int len) and free value after use)

# Setting Up Data For Python

## Setting Strings

- Null-terminate strings with char *namec = fix_string(name,*nlen)
- PyObject *pname = PyString_FromString(namec);
- PyObject *pMain = PyImport_AddModule("__main__");
- PyObject *pDict = PyModule_GetDict(pMain);
- PyDict_SetItemString(pDict,"n",pname);
- Py_DECREF(pname) when cleaning up to allow garbage collection by Python interpreter

## Setting integers/reals/doubles and objects

- char c[60]; // Allocate a string large enough for the command
- sprintf(c,"pynumber = %d", *myint);
- PyRun_SimpleString(c); //Execute the command

# (C) Passing It All In

## Creating a NumPy Array (see also Final Notes)

- int dims[] = {*size};
- PyObject *arr = PyArray_FromDimsAndData(
  - 1, - *dimensions*
  - dims, - *size in each dimension*
  - PyArray_DOUBLE, - *data type, alt. PyArray_INT*
  - (char*)data); - *pointer to the array*
- PyDict_SetItemString(pDict,"myPyArray",arr);

## (C) Creating the Object in Python

- char c[150+*slen+*mesh_name_len];
- // Create command with the variables we set
  sprintf(c,"field = ScalarField(n,myPyArray,ft,op);
  states['%s'].scalar_fields['%s'] = field", n, namec);
- PyRun_SimpleString(c); //Execute the command

# Python in Fluidity

# Passing Arrays to Python

## (F) python_add_array(real* myArray, string arrayName)

- Valid for 1-, 2- and 3-dimensional integer and real arrays
- E.g.: real, dimension(:,:,:) :: *myFortArray*
- python_add_array(*myFortArray*,"thisIsMyNameInPython")

## (C) python_add_array_double_2d(double* *myArray*, int **sx*, int **sy*, char* *arrayName*)

- Fortran wrapper functions as well as C standalone functions available in *python_statec.c*
- Functions for integers and doubles, 1d, 2d, 3d

# shape_dshape()

## (Py) Transform.shape_dshape()

- Equivalent to Fortran *shape_dshape()*
- Create Transform object to calculate Jacobian and Detwei
- Calculate dshape with *Transform.grad(coordinateField)*
- Call *shape_dshape(shape,dshape)*
- Example in *state_types.py*